### **Document Version and Revision History**

#### **Document Version: 1.1**

### **Date: 03.06.2024**

### **Note to Readers**

This document is a living document and may be subject to updates and revisions over time. Also not all elements of a project are presented in this document. Some parts of code responsible mainly for testing and graphical representation are not included in this document but will be attached to the main project report later on.

### **Connection Class**

The Connection class represents a connection between two neurons in adjacent layers. It stores the weight of the connection and the change in weight (delta weight) during training.

**Member Variables:**

* double weight: The weight of the connection which determines the strength of the connection between neurons.
* double deltaWeight: The change in weight of the connection during training which is used to update the weight during backpropagation.

**Member Functions:**

* Connection(): Constructor for the Connection class initializes the weight with a random value.
* double randomWeight(): Generates a random weight for the connection.

### **Layer Class**

The Layer class represents a layer of neurons in a neural network. It provides mechanisms for forward and backward propagation and gradient calculation.

**Member Variables:**

* std::vector<std::vector<Connection>> outputWeights: A vector storing the connections between neurons in this layer and neurons in the next layer.
* std::vector<double> m\_gradients: A vector storing the gradients for each neuron in the layer.
* std::vector<double> outputVals: A vector storing the output values for each neuron in the layer.
* static double eta: The overall net training rate.
* static double alpha: The multiplier of the last weight change (momentum).

**Member Functions:**

* size\_t size() const: Returns the number of neurons in the layer.

**Virtual Member Functions:**

* virtual void feedForwardLayer(const std::shared\_ptr<Layer>& prevLayer) = 0: Performs forward propagation.
* virtual void calcOutputGradients(const std::vector<double> &targetVals) = 0: Calculates gradients for the output layer.
* virtual void calcHiddenGradients(const std::shared\_ptr<Layer> &nextLayer) = 0: Calculates gradients for hidden layers.
* virtual void backPropagation(std::shared\_ptr<Layer> &prevLayer) = 0: Performs backpropagation.
* virtual double sumDOW(const std::shared\_ptr<Layer> &nextLayer, int n) const: Sums the derivatives of the weights.

### **LinearLayer Class**

The LinearLayer class is a derived class from Layer that uses a linear activation function.

**Member Functions:**

* void feedForwardLayer(const std::shared\_ptr<Layer> &prevLayer) override: Implements forward propagation for a linear layer.
* void calcOutputGradients(const std::vector<double> &targetVals) override: Implements gradient calculation for the output layer.
* void calcHiddenGradients(const std::shared\_ptr<Layer> &nextLayer) override: Implements gradient calculation for hidden layers.
* void backPropagation(std::shared\_ptr<Layer> &prevLayer) override: Implements backpropagation for a linear layer.

### **SigmoidLayer Class**

The SigmoidLayer class is a derived class from Layer that uses a sigmoid activation function.

**Member Functions:**

* void feedForwardLayer(const std::shared\_ptr<Layer> &prevLayer) override: Implements forward propagation for a sigmoid layer.
* void calcOutputGradients(const std::vector<double> &targetVals) override: Implements gradient calculation for the output layer.
* void calcHiddenGradients(const std::shared\_ptr<Layer> &nextLayer) override: Implements gradient calculation for hidden layers.
* void backPropagation(std::shared\_ptr<Layer> &prevLayer) override: Implements backpropagation for a sigmoid layer.
* double activationFunction(double x): Computes the sigmoid activation function.
* double activationFunctionDerivative(double x): Computes the derivative of the sigmoid activation function.

### **TanhLayer Class**

The TanhLayer class is a derived class from Layer that uses a tanh activation function.

**Member Functions:**

* void feedForwardLayer(const std::shared\_ptr<Layer> &prevLayer) override: Implements forward propagation for a tanh layer.
* void calcOutputGradients(const std::vector<double> &targetVals) override: Implements gradient calculation for the output layer.
* void calcHiddenGradients(const std::shared\_ptr<Layer> &nextLayer) override: Implements gradient calculation for hidden layers.
* void backPropagation(std::shared\_ptr<Layer> &prevLayer) override: Implements backpropagation for a tanh layer.
* double activationFunction(double x): Computes the tanh activation function.
* double activationFunctionDerivative(double x): Computes the derivative of the tanh activation function.

### **Net Class**

The Net class represents the entire neural network, which contains multiple layers of neurons.

**Member Variables:**

* std::vector<std::shared\_ptr<Layer>> layers: A vector storing the layers of the neural network.
* double error: The overall error of the network.
* double recentAverageError: The recent average error used for monitoring training progress.
* double recentAverageSmoothingFactor: The smoothing factor for computing the recent average error.

**Member Functions:**

* Net(const std::vector<unsigned> &topology): Constructor that initializes the network with the specified topology.
* void feedForward(const std::vector<double> &inputVals): Performs forward propagation through the network.
* void backProp(const std::vector<double> &targetVals): Performs backpropagation to update weights.
* void getResults(std::vector<double> &resultVals) const: Retrieves the output values from the output layer of the network.

Link to github: <https://github.com/Kokoszsz/neural-network>